1F−1　CASEデータ保存形式への設計記述言語MCLの採用

内金崎誠一，山岡芳樹，建部周二

(株)東芝　府中工場

1.はじめに

従来、ソフトウエア開発ツールの保存データは、データ入出力の性能、保存時の容量などの考慮により、バイナリ形式、あるいはツールに完全依存したテキスト形式が主流であった。近年普及のCASEに於いては、保存データは単なるファイルではなく、リポジトリとして保管され、さらにはCASEツール接続のためのデータ交換言語の標準化も検討されている。

東芝府中工場では、大規模リアルタイムソフトウエアの開発・生産を対象としたCASEを開発し、実プロジェクトで適用している。このCASEは、ツール開発と同時に、保存データ形式として、CASEを構成する複数のツール間で共通の構文を持ったテキスト形式の設計記述言語MCL(Module Components description Language)を開発、採用している。

本稿では、MCLの採用に関して、CASE開発という立場、および大規模リアルタイムソフトウエアの開発・生産へのCASE適用という立場に立ち、実用性という観点での評価を述べる。なお、本稿で述べるCASEは、EWSのASシリーズ上に開発し、社内での適用を経てパッケージソフトSEmateとして商品化されている。

2.MCLの概要

2.1 MCLの位置付け

SEmateにおけるMCLの位置付けを図1に示す。

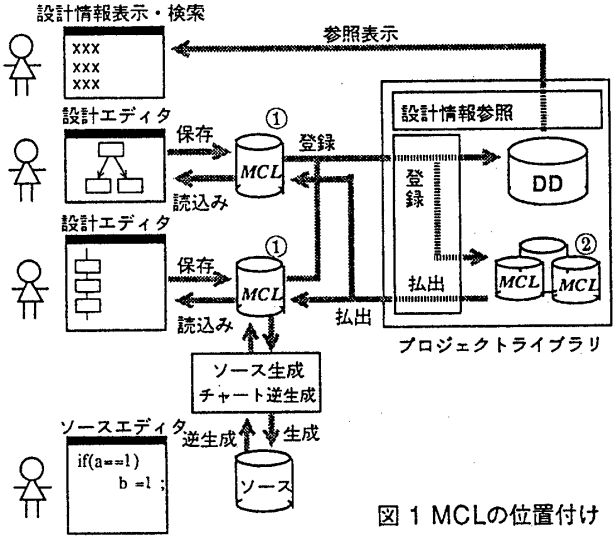

図1 MCLの位置付け

SEmateでは、各種設計エディタで作成中の設計シートは「シートアイコン」として表示し、設計が完了した設計シートは、プロジェクトライブラリ(SEmateのリポジトリ)に登録する。また、設計シートからはソースコードを生成したり、ソースコードから設計シートを逆生成するという機能を持つ。

MCLは、シートアイコンで表示される設計エディタで作成中の設計シートの保存データ、およびプロジェクトライブラリへの入力となる設計情報データ(図中①)である。プロジェクトライブラリに登録されたMCLは、設計成果物として管理される(図中②)。またソース生成/チャート逆生成の中間データとしても位置付けられる。なお、MCLが表す設計シート、設計情報は、次の表で示す中下流CASEの設計情報である。

表1 MCLが表す設計情報

設計情報の種類	設計シートの種類
タスク外部仕様	T-Spec：タスク仕様書
タスク内部仕様	V-MCD：タスク内モジュール構造図
パッケージ外部仕様	P-Spec：パッケージ仕様書
パッケージ内部仕様	P-MCD：パッケージ内モジュール構造図
モジュール外部仕様	M-Spec：モジュール仕様書
モジュール内部仕様	TFP / Flowchart / PAD
ライブラリ外部仕様	L-Spec：ライブラリ仕様書
データ仕様	F-Spec：ファイル仕様書 D-Spec：データ仕様書 S-Spec：データ構造仕様書

2.2 特徴

MCLは、設計情報の直接記述、データ交換を目的にした言語であり、次に示す特徴を持つ。

(1)構文解析が可能である

表1で示す設計シート毎に、記述される内容の詳細は異なるが、後述する理解性の確保、データ交換、およびその交換ツールの開発効率のため、設計シート間で共通、かつ構文解析が可能な文法を持ったテキスト形式のデータである。

(2)理解性が良い

常識的な英文知識、プログラミング知識があれば、MCLのコーディング方法など、特別なトレーニングを行わなくても理解できる。

(3)編集が可能である

通常のテキストエディタで編集が可能であり、通常のテキストファイルとして、UNIXの各種コマンドが使用できる。

(4)ソース生成/チャート逆生成を前提としている

モジュール内部仕様を表すMCLに於いては、C、Fortranを対象にした生成/逆生成を考慮し、C、Fortranの構文を網羅している。

Module components description language for CASE tools.

Seiichi UCHIKANEZAKI , Yoshiki YAMAOKA , Shuji TATEBE

TOSHIBA CORPORATION.

2.3 MCLの例

V-MCD(Vertical - Module Components connection Diagram：タスク内モジュール構造図)、フローチャートを例にMCLの構文を示す。

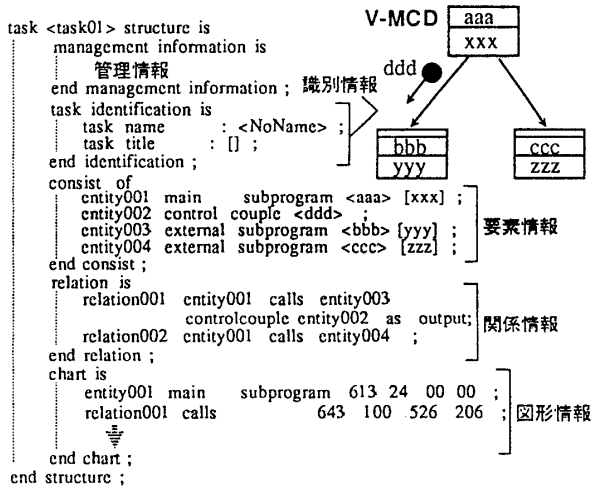

図 2 V-MCDのMCLの例

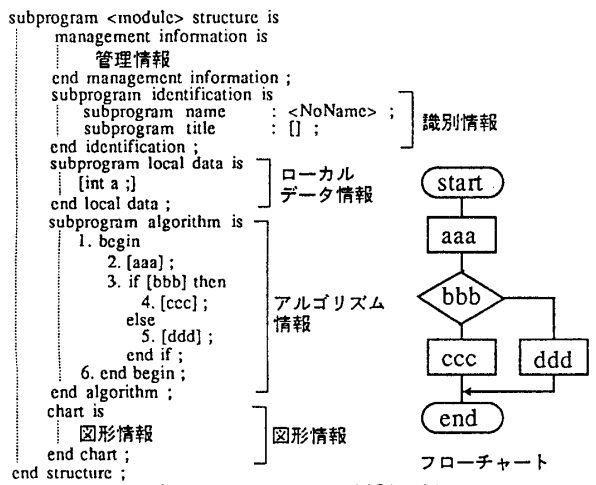

図 3 フローチャートのMCLの例

3.評価

SEmate開発、および実際の大規模リアルタイムソフトウエア開発・生産へSEmateを適用しての、MCLの主な評価を列挙する

3.1 CASE開発の立場で

(1)ツール開発の省力化

構文解析できる共通の言語形式であるので、その構文仕様をパーザ・ジェネレータのルールで定義することにより、各ツールで必要となるパーザ作成は大幅に省力化できる。また、構文仕様の追加・変更などの対応も容易である。

(2)チャート変換機能の実現

アルゴリズムチャートのMCLはチャートの種類に依存しない構文であるため、TFF、フローチャート、PADの間で自由にチャートの種類を変換する機能が実現できた。

3.2 CASE適用の立場で

(1)データ交換

他の開発ツールで作成した既存システムからの設計情報の流用を目的とした異なるCASE、開発ツール間での保存データのデータ交換が容易である。

(2)テキスト編集方式のインタフェース提供

一般的に、CASEは設計情報を設計図としてリポジトリへ格納するが、MCLにより、通常のテキストエディタを使用して、直接設計情報を記述・編集することができる。これは、設計情報入力の効率化、およびCASEを動作することができないPCなどの現有設備の有効利用など、実際の大規模システムの開発現場では非常に効果的である。

(3)耐障害性

システムに何等かの障害が発生し、データが正常に保存できなかった場合など、MCLを直接編集、修正することにより、リカバリできる。

4.課題

以下に課題を列挙する。

4.1 MCLの言語仕様に関して

(1)標準化

現在、IEEEより提案されているSTLなど、データ交換形式の標準化が検討されている。STLはおもに上流CASEの保存データを対象とし、MCLは中下流CASEを対象としているが、理解性、構造の解析が出来ることなど、考え方は同様である。今後、詳細仕様のレベルでSTLとの整合性を図りたい。

(2)拡張性

MCLは、基本的にツールに依存しない形式としているが、現実には、ツール固有の情報は存在する。ツール固有の情報の表現という意味での拡張、あるいはツールバージョンアップに伴い必要となってくる情報の追加・拡張ルールが、標準化や、異なるツールバージョン間での共通性の確保の観点から必要である。

4.2 ツール性能に関して

近年、計算機のパフォーマンスがかなり向上したとはいえ、MCLのパージングは処理全体のかなりのウエイトを占める。また、保存データの容量も大きくなる。小中規模プロジェクトへの適用では問題は無いが、大規模プロジェクトへの適用で、扱うデータ量が大きくなると、性能面で問題が生じる場合がある。ツール開発においてはパーザ、スキャナのチューニングなど、性能面での配慮が必要である。

5.まとめ

CASEの保存データを、直接記述、データ交換を目的に言語形式にし、実際のソフト開発の現場に適用した結果、その実用性はかなり高いと評価した。

今後は、適用でのノウハウをMCL仕様、ツール機能へ反映していくと同時に、今後国際的に標準化が予想される中下流CASEの保存データ形式に、現場でのノウハウを反映する様努めたい。

[参考文献]

[1]小野他 New-SWB大規模リアルタイム・ソフトウエア開発環境,情報処理学会第37回(昭和63年後期)全国大会3M-4

[2]藤本他 New-SWBプログラム設計支援ツール / MCD-tools,情報処理学会第37回(昭和63年後期)全国大会3M-7

[3]安田他 プログラム・ソースコード生成 / 設計書逆生成ツールの現場での適用と評価,情報処理学会第43回(平成3年後期)全国大会2K-13